1. 단계 설명

1. centos를 설치한 후, virtualenv를 다운받고 python2.7, mysql, flask를 설치하였습니다.
내부에 flaskapp이라는 virtualenv폴더를 생성하였고, flaskapp내부에 아래와 같은 app폴더를 생성하였습니다.

```
flaskapp/
        app/
         ├── cd.db
         ├── flaskr.py
         ├── read.me
         ├── schema.sql
         └── templates
             ├── index.html
             └── list.html
```

flaskapp 내부에 위와 같은 형태로 app을 구성하였습니다.
DB구축시 mysql을 이용하려 하였으나, 2002 socket error을 해결하지 못해 sqlite3을 이용하였습니다.

2. 사용자가 입력한 주식 종목 코드는 index.html에 k라는 값으로 받아, flaskr.py파일에서 작동하도록 하였습니다.
def search() 함수에서 k = request.args.get('k', "", type=str)를 통해 사용자가 입력한 주식종목코드를 받아왔고
html = urllib.urlopen("https://www.google.com/finance?q="+k+"&ei="+k) 이 코드를 이용해서 Google Finance에서 검색을 할 수 있게 하였습니다.

3. 받은 결과를
soup = BeautifulSoup(html, "html.parser")
price = soup.find('meta', attrs={'itemprop':'price'}) //현재가격
change = soup.find('meta', attrs={'itemprop':'priceChange'}) //가격변동 symbol = soup.find('meta', attrs={'itemprop':'tickerSymbol'}) //코드 exchange = soup.find('meta', attrs={'itemprop':'exchange'}) //나스닥, 코스피 …와 같은 종목
code = (exchange['content'])+":"+(symbol['content'])
time = str(datetime.now().strftime('%Y-%m-%d %H:%M:%S')) //검색한 당시 시간기록용
와 같이 파싱하였습니다.

4. 현재가격(price), 가격변동(price change) 두 가지 값을 결과로 보여주는 부분에는 json을 이용하여 생성하였습니다. search()함수의 return값으로
return jsonify(result="Price : "+ price['content'] +", Price Change : "+ change['content']) 을 주고
index.html에서

```
<script type=text/javascript>
$(function() {
```

```
  $('a#search').bind('click', function() {
          $.getJSON('/search', {
            k: $('input[name="k"]').val(),
          }, function(data) {
            $("#result").text(data.result);
          });
          return false;
        });
      });
 </script>
```

를 사용함으로써 사용자가 텍스트박스에 입력 후, search를 클릭하면 결과로 Price : 현재가격, Price Change : 가격변동 이 나올 수 있도록 하였습니다.

5. 하지만, 만약 google finance에 존재하지 않는 값을 입력한다면 code is not exist라는 값을 출력하도록 try except문을 search()함수 내에 사용하였습니다.

6. 보여준 값을 DB table에 저장하기 위해서 DB를 schema.sql 에 저장하였습니다.

```
drop table if exists searchlist;   //searchlist table이 존재하면 table 드랍
create table searchlist (
        no integer PRIMARY KEY AUTOINCREMENT, //DB입력순서
        code char(30) not null,   //주식종목코드
        time char(30) not null,   //검색한 당시의 시간
        price char(30) not null,   //현재 가격
        change char(30) not null);   //가격 변동 위와 같은 형태로 구성하였습니다.
```

DB명은 cd이고 DB를 초기화하고, 연결할때, 끝낼 때의 경우를 init_db(), get_db(), close_db()로 각각 구성하였습니다.

```
DATABASE = os.path.join(app.root_path, 'cd.db')
def init_db():
        with app.open_resource('schema.sql') as f:
                db.cursor().executescript(f.read())
        db.commit()

def get_db():
        if not hasattr(g, 'sqlite_db'):
```

```
            g.sqlite_db = sqlite3.connect(DATABASE)
        return g.sqlite_db

@app.teardown_request  def close_db(error):
        if hasattr(g,'sqlite_db'):
            g.sqlite_db.close()
```

터미널에 sqlite3 cd.db < schema.sql 을 입력하면 cd.db라는 DB가 생성이 됩니다. 이 cd라는 DB안에 존재하는 searchlist 테이블을 열어서 검색한 주식종목코드, 당시 시간, 당시 가격, 변동 가격을 입력을 해주기 위해서 db=get_db() 로 search() 함수 아래에서 DB를 연결하고 insert query를 실행하여 검색할 때마다 searchlist 테이블에 검색한 것을 입력해 줍니다.

```
db.execute('insert into searchlist (code, time, price, change) values (?, ?, ?, ?)', (code, time,
price['content'], change['content']))
```

db.commit()으로 변경사항을 반영해 줍니다.

DB에 저장된 기록을 불러내서 표로 보여주기 위해서 list()함수를 만들어 DB와 연결하고 select query를 실행하여서 searchlist의 모든 column들을 가져옵니다. searchlist에 튜플로 구성된 모든 정보를 입력하고 list.html로 연결해 줍니다.

```
def list():
        db = get_db()
        cur = db.execute('select * from searchlist')
        searchlist = cur.fetchall()
        return render_template('list.html', searchlist=searchlist)
```

list.html에는 아래와 같은 코드를 입력하여 Code, Time, Price, Price Change column을 표로 만들어 보여주기 위해 구성하였습니다.

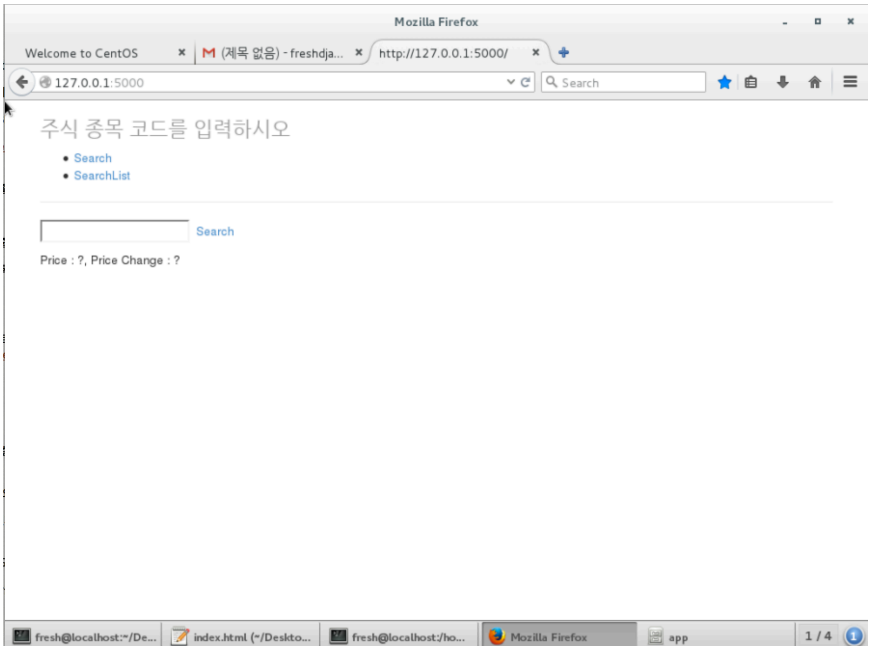
```
<ul>  <table><tr><th>Code</th><th>Time</th><th>Price</th><th>Price Change</th></tr>
        {% for slist in searchlist %}
        <tr><td>{{ slist[1] }} </td><td>{{ slist[2] }} </td><td>{{ slist[3] }} </td><td>{
{ slist[4] }}</td></tr>
        {% else %}
        <li><em>Unbelievable. No list! search </em>
        {% endfor %}</table></ul>
```

2. 소스 코드

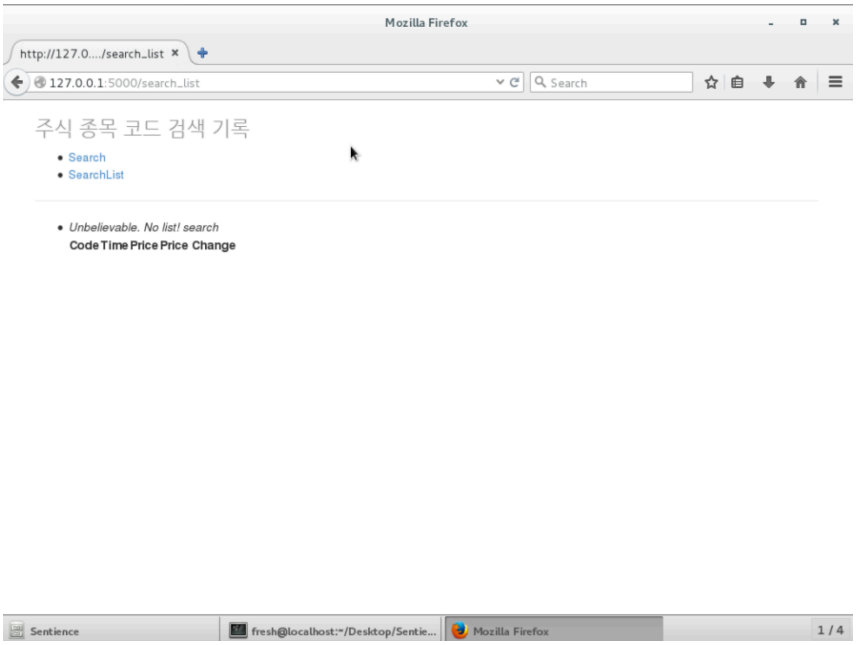
https://github.com/seonhyeshin/flask

3. 동작화면 스샷

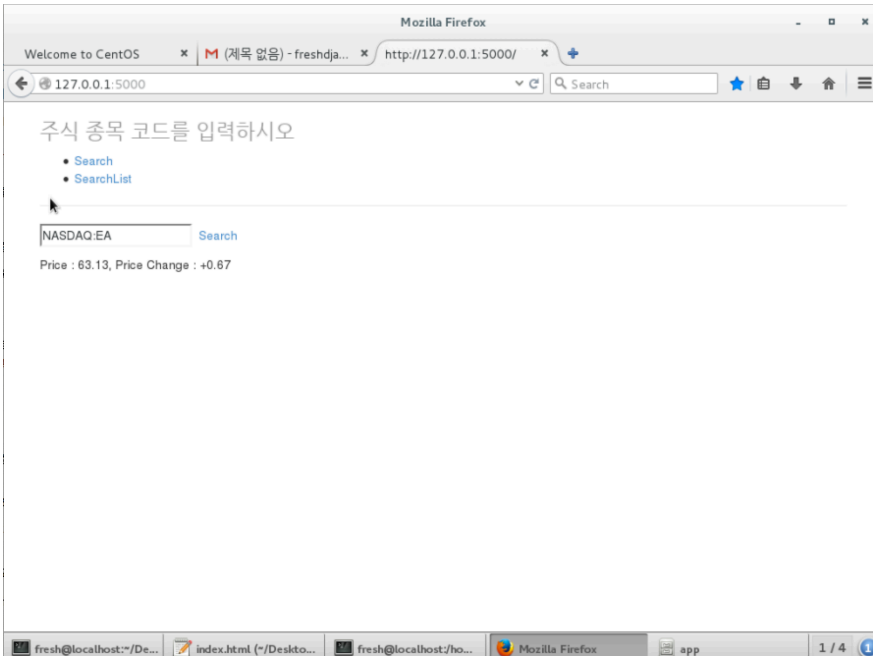
처음 실행 화면 index.html



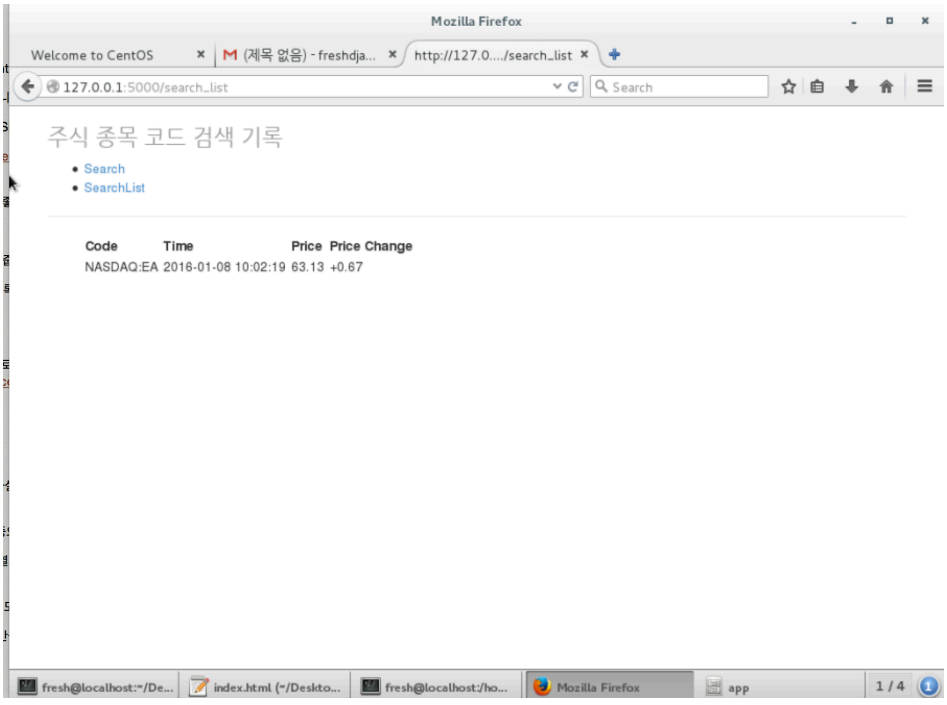
처음 실행 화면 list.html (아무것도 검색하지 않았을 때)



NASDAQ:EA 검색시(index.html)



NASDAQ:EA 검색시(list.html)



존재하지 않는 종목 코드 검색시(index.html)